INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Ordonnancement basé sur les réseaux de Petri

Feng CHU - Jean-Marie PROTH
Vanio Murilo SAVI

SAGEP

N° 1960
Juillet 1993

PROGRAMME 5

Traitement du signal,
automatique et
productique

Ordonnancement Basé sur les Réseaux de Petri

(Scheduling Using a Petri Net Model)

Feng Chu(*), Jean-Marie Proth(**) et Vanio Murilo Savi (*)

(*) INRIA-Lorraine, CESCOM, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE.

(**) INRIA-Lorraine, CESCOM, Technopôle Metz 2000, 4 rue Marconi, 57070 Metz, FRANCE et Institute for Systems Research, University of Maryland, College Park, MD 20742, USA.

Résumé. Les réseaux de Petri sont largement utilisés pour modéliser, spécifier et analyser les systèmes à événements discrets, en particulier les systèmes de production cycliques. Dans ce rapport, nous considérons un problème d'ordonnancement d'un système de production non cyclique. Les réseaux de Petri sont utilisés pour modéliser ce système. Le problème d'ordonnancement s'applique alors au franchissement des transitions. Nous proposons deux algorithmes : le premier est de type recuit simulé pour recherche d'un ordonnancement proche de l'optimum, le second est une procédure par séparation et évaluation pour trouver un ordonnancement optimal.

Mots Clés. Réseaux de Petri, Modélisation, Planification, Ordonnancement, Recuit Simulé, Procédure par séparation et évaluation.

Abstract. Petri Nets are widely used to model, specify and analyze discrete event systems. They are extensively studied in cyclic production environment. This paper addresses a scheduling problem for non cyclic production systems. Petri Nets are used to model these systems. In such a model the problem consists of scheduling the transition firing. We propose two algorithms to solve this problem: the first one is based on simulated annealing and applies to large size problems; the second one is a branch and bound method which is used for small and medium size problems.

Key Words. Petri Nets, Modeling, Planning, Scheduling, Simulated Annealing, Branch and Bound.

Introduction

Depuis la publication de la thèse de C.A. Petri dans les années 60 (Petri 1966), les réseaux de Petri ont reçu une attention qui ne cesse de croître. Ceci est dû au fait que les réseaux de Petri permettent de modéliser, spécifier et évaluer des systèmes à événements discrets (Murata 1977). Dans le domaine de la production, beaucoup d'efforts se sont portés sur l'application des réseaux de Petri à la modélisation et à l'évaluation des systèmes de production cyclique à l'aide de des graphes d'événements (Crétienne 1984, Hillion 1989, Hillion et Proth 1989, Laftit et al. 1992). Des contributions récentes montrent que les réseaux de Petri peuvent aussi modéliser un système de production non-cyclique et fournir une structure qui en facilite le contrôle (Harharlakis et al. 1992, Holloway et Krogh 1992, Krogh et Beck 1992).

Dans cet article, nous étudions le contrôle, plus particulièrement l'ordonnancement d'un système de production non cyclique. Pour cela nous utilisons les réseaux de Petri pour modéliser le système. Nous montrons que les problèmes de planification et d'ordonnancement peuvent être résolus à partir de ces modèles. Dans la littérature, qui reste très pauvre, on trouve quelques travaux sur les problèmes d'ordonnancement traités à l'aide des réseaux de Petri. Shih et Sekiguchi (1991) ont utilisé les réseaux de Petri pour modéliser un atelier flexible et ont proposé une méthode de "beam search" pour résoudre le problème d'ordonnancement. Lee et DiCesari (1992) ont considéré le problème d'ordonnancement qui consiste à minimiser la durée nécessaire pour produire un certain nombre de produits (makespan). Dans leur travail, les réseaux de Petri sont utilisés pour modéliser la polyvalence ainsi que le partage des ressources. Ils ont proposé un algorithme pour trouver le plus court chemin allant du marquage initial au marquage final. Ces marquages représentent respectivement le début et la fin de la production.

Dans cet article, nous nous limitons aux réseaux de Petri qui ont des transitions d'entrée et des transitions de sortie, en particulier les réseaux de Petri décomposables en CFIOs dont nous rappelons la définition dans la suite. Nous proposons deux algorithmes pour résoudre le problème d'ordonnancement : l'un est du type recuit simulé, l'autre est du type "procédure par séparation et évaluation". L'objectif de la planification est de calculer la quantité à fabriquer pour chaque type de produits durant chaque période élémentaire afin de minimiser la somme des coûts de stockage et des coûts de rupture. L'ordonnancement consiste à trouver le moyen de produire les quantités prescrites par la planification dans les meilleurs temps. On procède ensuite par horizon glissant.

Cet article est organisé en 5 sections. La section 1 est un rappel des définitions et des propriétés des réseaux de Petri. La section 2 donne les définitions des CFIOs, des CFIOs maximaux et des réseaux de Petri décomposables. Les résultats que nous présentons ensuite concernent les réseaux de Petri avec des transitions d'entrée et de sortie qui modélisent l'arrivée et le départ des produits du

système décomposable. La section 3 introduit simplement la planification comme une étape préliminaire de l'ordonnancement. La section 4 propose deux algorithmes d'ordonnancement et la section 5 présente des résultats numériques.

1. Rappels sur les réseaux de Petri

Un **réseau de Petri** est un graphe bi-parti dont les noeuds sont des places et des transitions. Les arcs relient les places aux transitions et les transitions aux places. Un réseau de Petri est souvent représenté par un quadruplet N = (P,T,A,W), où:

$P = \{p_1, p_2, ..., p_g\}$ est l'ensemble des places,

$T = \{t_1, t_2, ..., t_q\}$ est l'ensemble des transitions,

A est l'ensemble des arcs,

W est la valuation des arcs (valeurs positives entières).

Un réseau de Petri marqué est noté PN = <N, M> , où:

i) N est un réseau de Petri défini ci-dessus,

ii) $M : P \rightarrow N$ est le marquage de N, M(p) est le nombre de jetons dans la place p.

Si (t, p) (resp. (p, t)) $\in$ A, on dira que p est une place de sortie (resp. d'entrée) de la transition t et que t est une transition d'entrée (resp. de sortie) de la place p. On désigne par $t^\bullet$ (resp. $^\bullet t$) l'ensemble des places de sortie (resp. d'entrée) de la transition t, et par $p^\bullet$ (resp. $^\bullet p$) l'ensemble des transitions de sortie (resp. d'entrée) de la place p.

Une transition t est dite **franchissable** si et seulement si le nombre de jetons dans chacune de ses places d'entrée $p \in {}^\bullet t$ est supérieur ou égal à la valuation de l'arc (p, t). Le franchissement (ou le tir) d'une transition t consiste à enlever des jetons de chaque place d'entrée $p \in {}^\bullet t$ en nombre égal à la valuation de l'arc (p, t), et à ajouter des jetons dans chaque place de sortie $p' \in t^\bullet$ en nombre égal à la valuation de l'arc (t, p'). Ainsi, le marquage du réseau est modifié après le tir d'une transition.

Dans cet article, nous ne traitons que des réseaux à valuation unitaire.

En fait, le marquage d'un réseau représente l'état du système qu'il modélise. L'évolution du marquage représente donc l'évolution de l'état du système. C'est pourquoi les réseaux de Petri sont particulièrement appropriés pour la modélisation des systèmes dynamiques.

Si une séquence de transitions σ est franchissable pour un marquage M_0, alors nous arrivons à un autre marquage M après le tir des transitions de σ, et nous écrivons $M_0 \xrightarrow{\sigma} M$.

On peut montrer que le nouveau marquage est : $M = M_0 + C\overline{\sigma}$,

où $C = [c_{i,j}]$ est la matrice d'incidence avec, pour i = 1, 2, ..., g et j = 1, 2, ..., q,

$$c_{i,j} = \begin{cases} +1 & \text{si } p_i \in t_j^{\bullet} \\ -1 & \text{si } p_i \in {}^{\bullet}t_j \\ 0 & \text{sinon} \end{cases}$$

$\overline{\sigma}$ est le **vecteur caractéristique** de dimension q qui correspond à σ. La j-ième coordonnée correspondante représente le nombre d'occurrences de la transition t_j dans σ.

Un vecteur $V \geq 0$ est un **t-invariant** si et seulement si il existe un marquage M_0 et une séquence franchissable σ tels que $M_0 = M_0 + CV$ (i.e. $CV = 0$) avec $V = \overline{\sigma}$.

Dans un réseau de Petri, pour un t-invariant V donné, on peut construire un sous-réseau N_v appelé **sous-réseau du t-invariant** V tel que:

i) $t \in N_v$ si $t \in N$ et $v_t > 0$,

ii) $p \in N_v$ si $p^{\bullet} \cap T_v \neq \varnothing$ et ${}^{\bullet}p \cap T_v \neq \varnothing$.

où T_v est l'ensemble des transition dans N_v.

Un réseau de Petri est **consistent** s'il existe un marquage M_0 et une séquence σ franchissable tels que $M_0 \xrightarrow{\sigma} M_0$ et que chaque transition figure au moins une fois dans σ. Autrement dit, un réseau de Petri est consistent (resp. partiellement consistent) si et seulement si il existe un vecteur V positif (resp. non négatif) tel que $CV = 0$.

Dans cet article, nous n'étudions que les réseaux de Petri temporisés déterministes avec des transitions d'entrée et de sortie. Les transitions d'entrée et les transitions de sortie sont respectivement des transitions sans place d'entrée et des transitions sans place de sortie.

2. Décomposition d'un modèle de réseaux de Petri en CFIOs maximaux

Dans ce qui va suivre, nous utilisons un réseau de Petri avec transitions d'entrée et de sortie pour modéliser un système ou un sous-système de production. Les transitions d'entrée et les transitions de sortie modélisent respectivement l'arrivée des matières premières (ou des produits semi-finis) et le départ des produits finis (ou semi-finis). Il se peut que ces modèles comportent des conflits structurels, c'est-à-dire qu'une place ait plusieurs transitions de sortie (voir la figure 1). Nous devons décider lesquelles de ces transitions franchir si le nombre de jetons dans la place est inférieur au nombre de ses transitions de sortie.

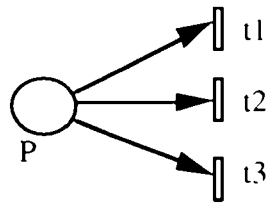

Fig. 1. Conflit Structurel

Comme nous l'avons soulingné plus haut, un modèle de réseau de Petri peut comporter des conflits structurels. Pour gérer ces conflits, nous décomposons le modèle en CFIOs (Conflict Free nets with Input and Output transitions) dans lesquels il n'y a pas de conflit structurel et qui possèdent toutes les bonnes propriétés, c'est-à-dire qu'ils sont vivants, bornés et consistents (Harharlakis et al. 1992). Nous rappelons ici la définition d'un CFIO.

Un réseau de Petri sans conflit avec des transitions d'entrée et de sortie (CFIO) est tel que:

i) il est sans conflit structurel:

$$^\bullet t_i \cap {}^\bullet t_j = \varnothing, \quad \forall t_i, t_j \in T$$

ii) il n'y a pas de places sources ni de places puits:

$$^\bullet T = T^\bullet$$

iii) chaque place a au moins une transition d'entrée et une transition de sortie:

$$^\bullet P - (^\bullet P \cap P^\bullet) \neq \varnothing \quad \text{et} \quad P^\bullet - (^\bullet P \cap P^\bullet) \neq \varnothing$$

Pour qu'un réseau NC soit un CFIO du réseau de Petri N avec des transitions d'entrée et de sortie, il faut que:

i) NC soit un CFIO connexe;

ii) les transitions d'entrée (resp. de sortie) de NC soient des transitions d'entrée (resp.de sortie) de N;

iii) un noeud (place ou transition) de NC soit un noeud de N;

iv) pour tout $t \in$ NC, l'ensemble $\bullet t$ (resp. $t\bullet$) dans NC soit le même que dans N.

Dans un travail antérieur (Chen et al. 1993), nous avons défini un CFIO maximal et avons proposé une méthode pour décomposer un réseau de Petri avec transitions d'entrée et de sortie en CFIOs.

NC est appelé CFIO maximal de N, s'il n'existe pas d'autre CFIO de N, NC*, tel que NC $\subset$ NC*.

Un réseau de Petri avec transitions d'entrée et de sortie est **décomposable** si :

$$N = \bigcup_{i \in C} NC_i$$

où $C = \{i / NC_i$ est un CFIO maximal de N et NC_i est consistent$\}$.

Un réseau de Petri décomposable a les propriétés suivantes:

i) il est consistent;

ii) il peut être maintenu borné en contrôlant les transitions d'entrée et de sortie;

iii) il est vivant.

3. Planification basée sur les CFIOs maximaux

Si un réseau de Petri N avec transitions d'entrée et de sortie modélise un système ou un sous-système de production, et que N est décomposable, on dit que le système est contrôlable. On considère NC_1, NC_2, ..., NC_m, les CFIOs maximaux connexesdu réseau de Petri N. Pour tout k=1, 2, ..., m, NC_k est un sous-réseau du t-invariant $V_k = \left[v_1^k, \ldots, v_q^k\right]$ (voir section 1 pour la définition).

Au niveau de la planification, nous cherchons à minimiser le coût total de rupture et de stockage, f(A), où $A = [a_{i,k}]$, i = 1, 2,..., h; k = 1, 2,...m est la variable de décision.

Le problème s'écrit:

$$\text{Minimiser } f(A) = \sum_{j=1}^{n}\left[p_j^+ \sum_{s=1}^{h}\left[\sum_{i=1}^{s}\left(x_{i,j}-d_{i,j}\right)\right]^+\right] - \sum_{j=1}^{n}\left[p_j^- \sum_{s=1}^{h}\left[\sum_{i=1}^{s}\left(x_{i,j}-d_{i,j}\right)\right]^-\right]$$

où

$a^+ = \mathit{Max}(a,0)$ et $a^- = \mathit{Min}(a,0)$,

h est le nombre de périodes élémentaires,

n est le nombre de types de produits,

$d_{i,j}$ est la demande du produit de type j à la fin de la période i,

p_j^+ est le coût de stockage d'une unité de produit de type j pendant une période élémentaire,

p_j^- est le coût de rupture d'une unité de produit de type j pendant une période élémentaire,

$x_{i,j}$ est le nombre de produits de type j à fabriquer pendant la période élémentaire i,

$$x_{i,j} = \sum_{t \in T_j}\left(\sum_{k=1}^{r_j} a_{i,k} v_t^k\right) \text{ pour i = 1, 2,..., h et j = 1, 2, ..., n,}$$

où :

$a_{i,k}$ est le nombre de fois où le CFIO maximal NC_k est mis en fonction,

T_j est l'ensemble des transitions de sortie correspondant au produit de type j,

r_j est le nombre de CFIOs maximaux connexes concernant le produit de type j.

Notons que des CFIOs non maximaux et connexes peuvent être utilisés. Dans ce cas, les résultats ne peuvent être que meilleurs, mais les calculs sont plus volumineux.

La contrainte suivante doit être respectée :

$$\sum_{t \in Q(R)} \left(\sum_{k=1}^{m} a_{i,k} v_t^k \right) \theta_t \leq u \text{ pour } i = 1, 2, ..., h \text{ et } R = 1, 2, ..., M.$$

où:

Q(R) est l'ensemble des transitions correspondant à la ressource R,

θ_t est le temps de franchissement de la transition t,

u est la durée d'une période élémentaire,

M est le nombre de ressources.

Cette contrainte exprime le fait que la charge totale de chaque ressource doit être inférieure ou égale à la durée d'une période élémentaire.

Nous voyons que nous cherchons à minimiser le coût total en franchissant des ensembles des transitions, chacun de ces ensembles correspondant à un CFIO maximal. Nous introduisons donc une rigidité dans le système : c'est le prix à payer pour s'assurer des "bonnes" propriétés qualitatives.

Nous proposons un petit exemple qui représente un processus de fabrication concernant 3 types de produits, 5 machines et 5 périodes élémentaires.

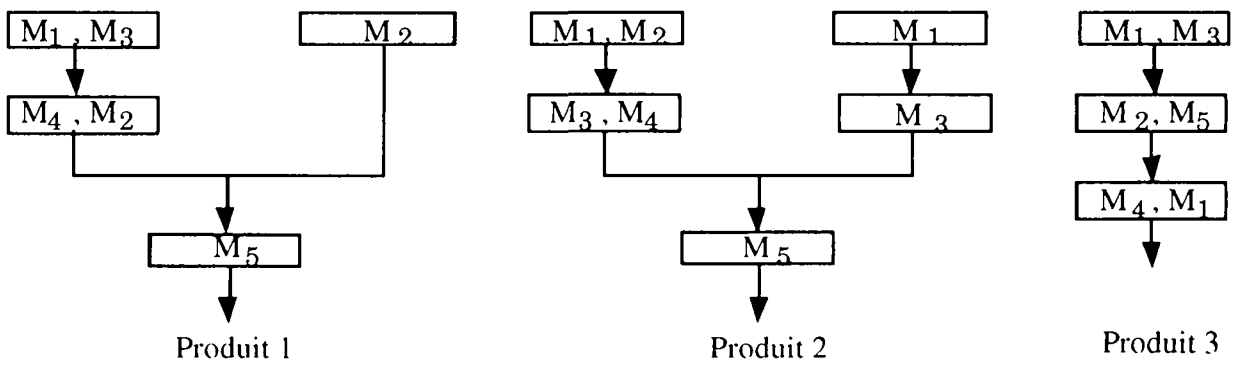

Fig. 2. Gammes de fabrication

La figure 2 représente la gamme des trois types de produits. Chaque boîte correspond à une opération et la gamme de fabrication contient une liste des machines capables d'effectuer cette opération.

La figure 3 schématise le modèle de réseaux de Petri correspondant à ces gammes. Chaque transition représente une opération effectuée sur une ressource sauf celles dont le temps de franchissement est nul. Celles-ci ne représentent que le début de production.

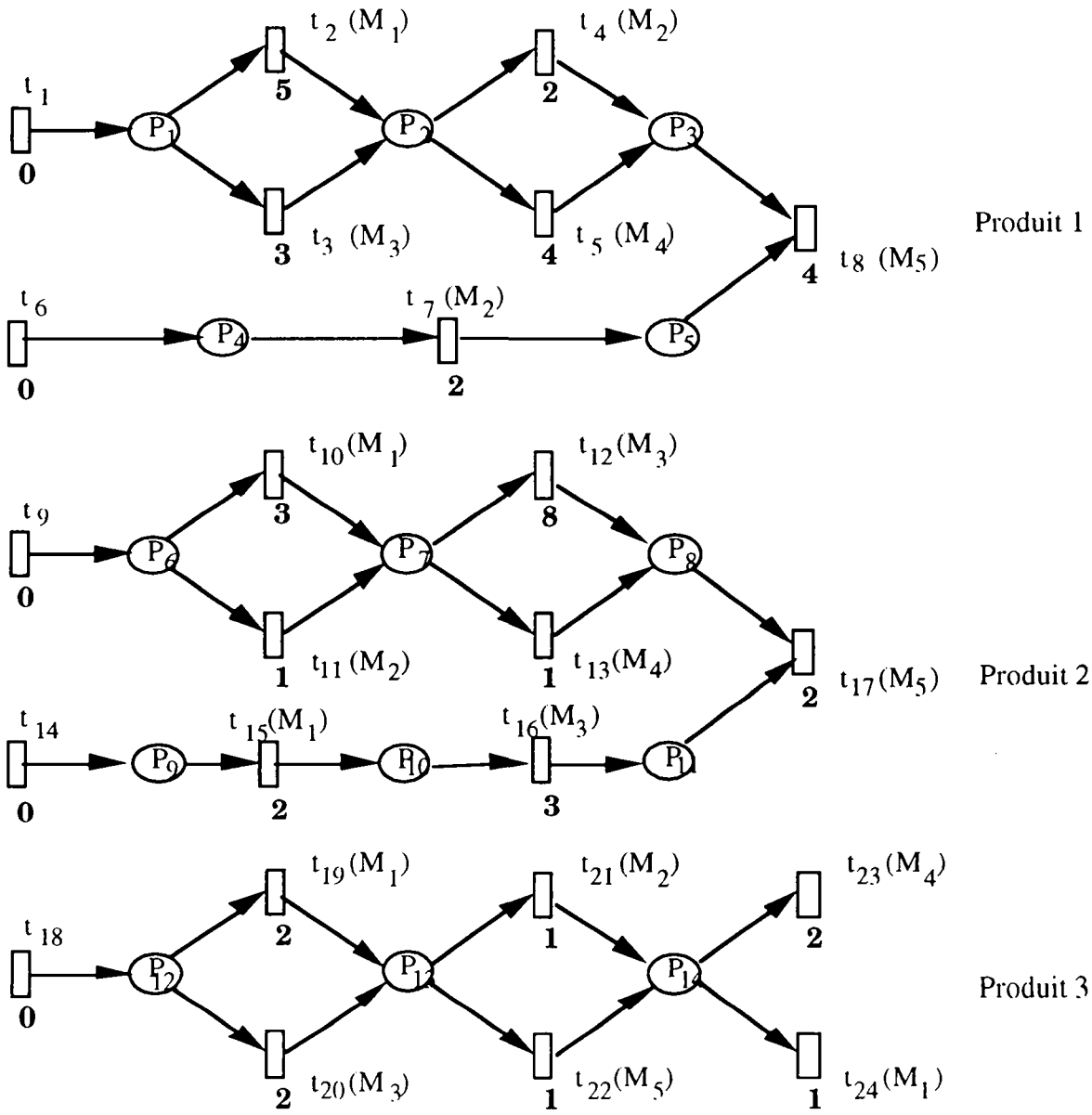

Fig. 3. Modèle des gammes de fabrication

La durée d'une période élémentaire réelle est égale à 80. Nous choisissons la durée d'une période élémentaire artifitielle égale à 70 (on prévoit une utilisation des ressources à moins de 100% pour que la solution de la planification soit réalisable au niveau de l'ordonnancenemt).

Nous supposons qu'au début il n'y a pas de stock, et que toutes les demandes doivent être satisfaites à la fin de la 5-ième période.

Les demandes pour les trois types de produits pendant les 5 périodes sont données dans la table 1.

Table 1. Demandes

Périodes ⇒ Produits ⇓	1	2	3	4	5
1	12	10	11	12	12
2	12	14	15	8	9
3	20	9	12	11	8

Les coûts de stockage et de rupture pour une période élémentaire pour les 3 types de produits sont fournis par la table 2.

Table 2. Coûts de stockage et de rupture

Périodes ⇒ Coûts ⇓	1	2	3
p_j^+	10	20	30
p_j^-	30	20	10

Dans cet exemple, les productions ne sont pas égales aux demandes à la fin de certaines périodes, mais à la fin de la 5-ième période la production totale est égale à la demande totale, comme le montre la table 3.

Table 3. Comparaison des demandes (D) et des productions (P)

Produits ⇒	1		2		3	
Périodes⇓	D	P	D	P	D	P
1	12	12	**12**	**11**	20	20
2	10	10	**14**	**15**	9	9
3	**11**	**10**	15	15	12	12
4	**12**	**13**	8	8	**11**	**14**
5	12	12	9	9	**8**	**5**

Les résultats de la planification nous permettent d'obtenir $a_{i,k}$, le nombre de fois où le CFIO maximal NC_k doit fonctionner pendant la i-ème période élémentaire. Par conséquent, nous

connaissons le nombre de franchissements d'une transition t, qui est égal à $\sum_{k=1}^{r_j} a_{i,k} v_t^k$. Ces nombres de franchissement des transitions sont représentés dans la table 4.

La planification nous fournit des informations très importantes, comme les nombres de franchissements des transitions, la charge de la ressource goulot pendant chaque période, etc. Ces informations sont utilisées dans l'ordonnancement.

Table 4. Nombres de franchissements des transitions

Transitions ⇒ Périodes ⇓	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
1	12	3	9	7	5	12	12	12	11	0	11	0
2	10	7	3	10	0	10	10	10	15	1	14	0
3	10	3	7	5	5	10	10	10	15	0	15	0
4	13	7	6	12	1	13	13	13	8	3	5	1
5	12	8	4	12	0	12	12	12	9	1	8	1

Transitions ⇒ Périodes ⇓	t13	t14	t15	t16	t17	t18	t19	t20	t21	t22	t23	t24
1	11	11	11	11	11	20	15	5	20	0	17	3
2	15	15	15	15	15	9	1	8	9	0	9	0
3	15	15	15	15	15	12	10	2	12	0	11	1
4	7	8	8	8	8	14	4	10	12	2	12	2
5	8	9	9	9	9	5	3	2	2	3	2	3

4. Ordonnancement

A la fin de la planification, nous connaissons le nombre de franchissements des transitions sur chaque période élémentaire. Mais nous n'avons pas considéré les éventuels conflits structurels et le fait que les transitions concernant une même ressource ne peuvent pas être franchies en même temps. En plus, au niveau de la planification, nous ne savons pas comment la production va s'organiser : c'est le rôle de l'ordonnancement. L'ordonnancement sera un succès si nous parvenons à organiser le système de sorte que les franchissements prévus au cours de la première période élémentaire puissent effectivement être réalisés.

Le modèle que l'on considère en ordonnancement est dérivé de celui de la planification (voir la figure 4). On ajoute une place pour chaque ressource. Cette place contient un seul jeton. Elle est à la fois place d'entrée et place de sortie des transitions qui représentent une opération effectuée sur la ressource pour modéliser la contrainte qui dit que la ressource peut effectuer au plus une opération à la fois.

Au niveau de l'ordonnancement, nous ne considérons que la première période. Reprenons l'exemple introduit dans la section 3. Nous introduisons cinq places notées Q_1, Q_2, Q_3, Q_4 et Q_5 correspondant respectivement aux M_1, M_2, M_3, M_4 et M_5. Pour simplifier, dans la figure 4 nous ne représentons que la place Q_1.

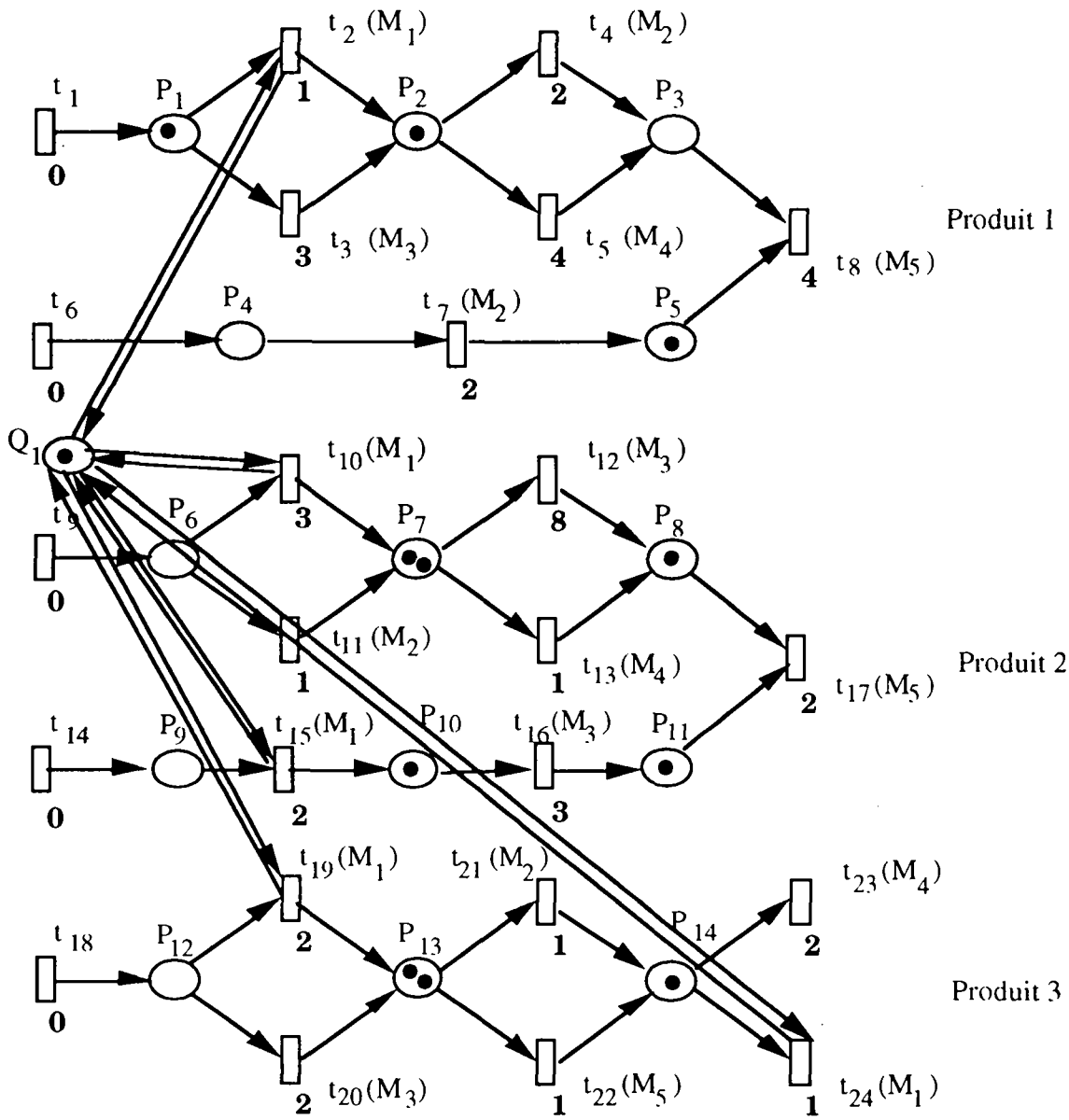

Fig. 4. Modèle pour l'ordonnancement

Nous supposons que le marquage du système au début de la première période M_0 est le suivant :

$M_0 = < 1,1,0,0,1,0,2,1,0,1,1,0,2,1,1,1,1,1,1 >$,

et nous ferons l'hypothèse que les transitions sont franchies dès qu'elles sont franchissables. Nous montrons maintenant comment le problème d'ordonnancement se résoud pour réaliser les nombres de franchissements des transitions.

Le problème d'ordonnancement est un problème NP-difficile [French 1982]. Nous développons une méthode de recuit-simulé pour les problèmes de grande taille, et une méthode de type "procédure par séparation et évaluation" pour les problèmes de petite taille. Dans cet article, nous comparons les performances de ces deux approches. Les deux paragraphes suivants expliquent l'implémentation de ces deux algorithmes.

4.1. Recuit Simulé

L'approche du recuit simulé est une approche de recherche par voisinage. Comme avantages, elle permet d'éviter de tomber dans un optimum local et fournit une solution réalisable tout au long du déroulement de l'algorithme. Si l'utilisateur est satisfait de la meilleure solution trouvée, il peut arrêter le déroulement de l'algorithme.

Dans cette approche nous utilisons les variables de contrôle suivantes :
T_0 — température initiale;
T_f — température finale;
N_a — nombre maximal de solutions acceptées pour un palier de température;
N_r — nombre maximal de solutions rejetées pour un palier de température;
ΔT — raison géometrique de la diminution de température.

L'algorithme utilisé s'écrit:

1. Donner une température initiale $T := T_0$. Générer au hasard une solution initiale x_0 (cette solution peut aussi être obtenue en utilisant une heuristique).

2. Accepter x_0 comme la meilleure solution trouvée : $x^*=x_0$, Nb_accept :=0, Nb_rejet :=0.

3. Générer au hasard une solution x_1 dans le voisinage de x_0.

4. Si $f(x_1) \leq f(x_0)$ alors accepter la solution x_1 ($x_0 := x_1$, Nb_accept := Nb_accept + 1) sinon accepter x_1 ($x_0 := x_1$, Nb_accept := Nb_accept + 1) avec une probabilité $\exp\{(f(x_0) - f(x_1))/T\}$.

5. Si $f(x_0) < f(x^*)$, x_0 devient la meilleure solution trouvée : $x^* := x_0$.

6. Si Nb_accept > N_a ou Nb_rejet > N_r alors $T := T^* \Delta T$, Nb_accept :=0, Nb_rejet :=0.

7. Si $T \leq T_f$ alors éditer x* comme la meillure solution et STOP, sinon aller à 3.

Pour calculer la valeur du critère (makespan) d'une solution x, nous faisons une simulation.

Comme on peut le constater, le problème consiste à définir une solution et à définir un voisinage d'une solution. Nous l'avons dit plus haut, après le niveau de planification, nous connaissons le nombre de franchissements pour chaque transition. Mais toutes les transitions ne peuvent pas être franchies en même temps, en raison de la présence des conflits structurels et du fait qu'une ressource peut effectuer au plus une opération à la fois. Il faut décider l'ordre d'affectation des jetons aux transitions de sortie pour toute place comportant plusieurs transitions de sortie. Par exemple, si t1 et t2 sont deux transitions de sortie d'une place p, la séquence (t1, t1, t2,...) pour la place p signifie que les deux premiers jetons arrivés dans p sont destinés à tirer deux fois la transition t1, le troisième jeton permet le franchissement de la transition t2 et ainsi de suite.

Dans notre problème, deux types de décisions sont à prendre : l'une concerne le choix d'une ressource pour effectuer une certaine opération si plusieurs ressources sont possibles (appelée CR : Choix Ressource); l'autre est le choix d'une opération à effectuer sur une ressource si celle-ci concerne plusieurs opérations (appelée CO : Choix Opération). Une solution consiste à choisir une séquence pour toute place représentant une resource (séquence CO) et une séquence pour toute place représentant une opération et qui a plusieurs transitions de sortie (séquence CR).

Si on génère les deux types de séquences CR et CO indépendemment, on risque de bloquer le système. Au titre d'exemple, considérons le cas illustré par la figure 5 où on a 6 transitions et 2 places avec conflit. Supposons que le temps de franchissement de chaque transition soit une unité. La transition t1 concerne la ressource 1 et doit être franchie 3 fois. Le marquage initial est $M_0 = (0, 0)$.

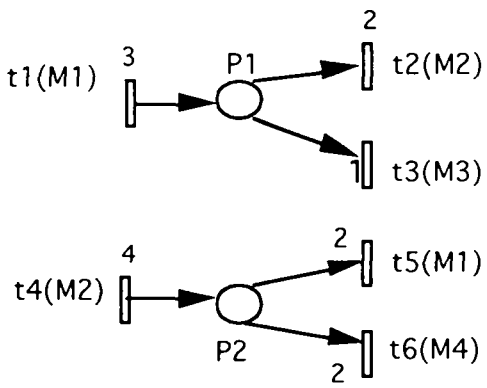

Fig. 5. Exemple de blocage

Nous montrons que si on génère les séquences CR et CO comme suit, le système sera bloqué.

CO: (M1) t5,t1,t5,t1,t1;

(M2) t2,t4,t4,t2,t4,t4.

CR: (P1) t2,t3,t2;

(P2) t5,t6,t5,t6.

En effet, si on veut franchir t5, il faut d'abord franchir t4 car sinon il n'y a pas de jeton dans la place d'entrée de la transition t5 pour la franchir. En tenant compte de la séquence CO, le franchissement de t4 ne peut avoir lieu qu'après celui de t2 alors que celui-ci ne peut avoir lieu qu'après le franchissement de t1 du fait que le marquage de P1 est nul. Comme on peut le constater, ceci n'est possible qu'après le franchissement de t5. On entre ainsi dans une situation de blocage.

Par contre, si on génère seulement la séquence CR et que la séquence CO est construite au fur et à mesure de la simulation cette situation de blocage peut être évitée. C'est pourquoi dans notre algorithme nous générons les séquences CR, et en se basant sur CR nous construisons les séquences CO pendant la simulation.

Pour ce qui concerne le voisinage d'une solution, nous décidons de modifier la séquence de 20% des places CR, tirées au hasard. Cette modification consiste à permuter au hasard deux transitions différentes.

Le désavantage d'un algorithme de type recuit-simulé est qu'on ne peut jamais assurer l'optimalité de la solution trouvée sauf dans le cas où la ressource goulot est saturée. En plus, même si la taille du modèle est petite, le temps de calcul est relativement long, alors que la solution n'est pas nécessairement bonne.

4.2. Procédure par séparation et évaluation

La procédure par séparation et évaluation consiste à décomposer successivement un sous-ensemble de solutions en des sous-ensembles encore plus petits. Pour chaque sous-ensemble obtenu, on calcule une borne inférieure de toute solution de ce sous-ensemble. Si cette borne est supérieure ou égale à la valeur du critère ("makespan") de la meilleure solution déjà trouvée, on est sûr qu'il n'existe pas de solution améliorante dans ce sous-ensemble. Par conséquent, on abandonne la recherche dans ce sous-ensemble. Cette approche est souvent représentée par une arborescence où chaque nœud représente un sous-ensemble. Dans la suite du paragraphe, nous parlons de "noeuds", au lieu de sous-ensemble de solutions. Les nœuds descendants d'un nœud sont les sous-ensembles obtenus en décomposant les sous-ensembles représenté par celui-ci.

Dans cette méthode, nous devons disposer d'une borne supérieure qui est la valeur du critère de la meilleure solution déjà trouvée. Nous utilisons la durée réelle d'une période élémentaire comme la borne supérieure initiale.

Dans notre algorithme, chaque nœud représente une solution partielle où au moins une transition commence à être franchie à l'instant τ. Dans cette solution partielle, on connaît le nombre de franchissements restants σ_t pour chaque transition t. On a également un ensemble de transitions en cours de franchissement F. Pour chaque transition $t \in F$, on connaît le temps de franchissement restant à l'instant τ, soit θ'_t. Si on désigne par T_k l'ensemble des transitions représentant les opérations effectuées par la ressource k, alors il n'existe pas deux transitions i et j telles que $i \in T_k$, $j \in T_k$, $i \in F$ et $j \in F$ (la ressource k effectue au plus une opération à la fois). Autrement dit, $\text{card}(F \cap T_k) \leq 1$. Comme nous l'avons souligné plus haut, il faut associer à chaque noeud une borne inférieure. Pour cela nous introduisons une quantité φ_k pour toute ressource k (k = 1, 2, ..., M) telle que

$$\varphi_k = \begin{cases} \theta'_t & \text{si } t \in F \cap T_k, \\ \min\limits_{t \in F} \theta'_t & \text{si } F \cap T_k = \emptyset. \end{cases}$$

En effet, si la ressource k est occupée à l'instant τ, alors, φ_k représente la durée de la période pendant laquelle la ressource k continue à être occupée. Par conséquent, elle ne peut commencer à effectuer une nouvelle opération qu'à partir de l'instant $\tau+\varphi_k$. Si la ressource k est inoccupée à l'instant τ, alors aucune des transitions appartenant à T_k n'est franchissable (toutes les places d'entrée ne contiennent pas suffisamment de jetons). Il faut que le franchissement d'au moins une transition soit achevé pour qu'il soit possible qu'une des transitions appartenant à T_k devienne franchissable. Cette durée est précisément représentée par φ_k. Dans touts les cas, $\tau+\varphi_k$ représente l'instant le plus petit où une des transitions appartenant à T_k devienne franchissable.

En outre, même si la ressource k peut commencer à effectuer une opération à l'instant $\tau+\varphi_k$, l'ordonnancement ne peut être terminé avant que la ressource k n'ait effectué sa charge restante. Celle-ci peut être minorée par $\sum_{t \in T_k} \sigma_t \times \theta_t$. Le makespan est donc supérieur ou égal à $\tau+\varphi_k+\sum_{t \in T_k} \sigma_t \times \theta_t$, pour k=1, 2, ..., M. Autrement dit, la quantité

$$LB=\tau+\max_{k=1,2,\ldots,M}\left\{\sum_{t \in T_k} \sigma_t \times \theta_t + \varphi_k\right\}$$

est une borne inférieure du makespan, compte tenu de la solution partielle représentée par le nœud.

Lors de la séparation d'un nœud, nous créons des nœuds descendants après la fin du franchissement de la (ou des) première(s) transition(s) en cours de franchissement dans le nœud ascendant, en considérant toutes les combinaisons possibles des franchissements des transitions.

Nous identifions tout d'abord l'ensemble des transitions franchissables Φ_k pour chaque ressource libre k. Une transition est franchissable si et seulement si chacune de ses places d'entrée contient au moins un jeton. On crée autant de nœuds qu'il y a de combinaisons possibles en prenant un élément de chaque Φ_k. Bien entendu, ces nœuds ne sont pas tous réalisables en raison de conflits structurels. Si deux transitions $t \in F_k$ et $t' \in F_{k'}$ partagent une même place d'entrée qui ne contient qu'un seul jeton, alors t et t' ne peuvent pas être franchies en même temps. Dans ce cas, ce nœud est remplacé par deux nouveaux nœuds tels que dans l'un, t est franchie mais pas t' et dans l'autre t' est franchie mais pas t.

Dans un nœud descendant, l'ensemble des transitions en cours de franchissement est constitué des nouvelles transitions dont le franchissement vient de commencer et des transitions du nœud ascendant dont le franchissement n'est pas encore achevé.

En ce qui concerne le choix du nœud à séparer, nous utilisons une méthode en profondeur d'abord modifiée : parmi les nœuds créés le plus récemment, l'un de ceux qui ont la plus petite borne inférieure est choisi pour la séparation suivante. Un choix plus fin dans ce cas permettrait d'accélérer la recherche. Lorsque deux nœuds ont la même borne inférieure, on choisit le nœud contenant le plus de transitions dont une des places de sortie n'a pas de jeton. Ceci permettra d'accélérer le flux des jetons. En cas d'égalité, on choisit un nœud contenant une transition dont le nombre de jetons dans la place d'entrée est le plus petit. Ce choix est justifié par le fait que si cette transition est tirée, alors la place d'entrée devient vide plus rapidement et par conséquent à la prochaine séparation, une des transitions d'entrée de cette place aura plus de chance d'être tirée.

Le désavantage d'une méthode de type "procédure par séparation et évaluation" est que le besoin en mémoire et le temps de calcul augmente très vite avec la taille du problème à résoudre. C'est pourquoi l'application de ce type de méthode se limite à des problèmes de taille raisonnable.

5. Résultats numériques

Pour comparer les performances des algorithmes, nous générons des exemples sur lesquels nous appliquons les deux algorithmes. Nous choisissons quatre modèles de réseaux de Petri avec des transitions d'entrée et des transitions de sortie. Le premier modèle est l'exemple introduit dans la section 3. Ces modèles sont consistents et décomposables. La table 1 résume les caractéristiques de ces modèles.

Table 5. Caractéristiques des modèles

Exemple	nombre de types de produits	nombre de ressources	nombre de places	nombre de transitions
1	3	5	14	24
2	5	5	15	28
3	5	5	30	46
4	8	5	42	66

Pour chacun de ces modèles, nous générons au hasard la demande de produits de chaque type. Il faut noter que cette demande peut ne pas être réalisable. Après la planification, on obtient la quantité réelle à fabriquer pour chaque type de produits pendant chaque période élémentaire. Ces quantités se traduisent par les nombres de franchissements des transitions. Les expériences numériques ont été effectués sur un IBM RISC 6000.

Les tableaux suivants présentent les résultats numériques pour chacun des quatre modèles. A partir de ces tableaux on peut comparer les makespans trouvés et les temps de calcul pour chacun des deux algorithmes, où:

NF est le nombre total de franchissements des transitions;

NP est le nombre total de produits;

RS concerne l'algorithme de recuit simulé;

BAB concerne l'algorithme de la procédure par séparation et évaluation;

m est le makespan;

t est le temps de calcul (en seconde du CPU).

Dans la méthode de recuit simulé, nous avons utilisé les paramètres de contrôle suivants : $T_0 = 100$, $T_f = 0{,}01$, $N_a = 5$, $N_r = 5$, $\Delta T = 0{,}92$.

Table 6. Comparaison des résultats pour le modèle 1

NF	NP	RS(m)	BAB(m)	RS(t)	BAB(t)
32	8	15	15	5,04	0,31
75	19	30	30	4,96	0,48
162	43	73	70	25,20	0,71
289	76	126	120	46,23	1462,42*
297	79	127	120	47,68	1,40
385	100	167	160	54,04	32,86
485	127	208	200	82,34	1,83
745	188	324	320	120,56	3,54

Table 7. Comparaison des résultats pour le modèle 2

NF	NP	RS(m)	BAB(m)	RS(t)	RS(t)
35	18	20	20	6,17	0,10
52	27	39	39	8,49	0,13
134	69	90	90	30,62	0,34
220	117	120	120	47,8	0,85
263	135	143	140	67,79	0,85
289	151	160	160	60,03	0,78
392	206	200	200	63,54	3,46
593	335	300	300	139,01	2,55

Table 8. Comparaison des résultats pour le modèle 3

NF	NP	RS(m)	BAB(m)	RS(t)	BAB(t)
58	16	30	30	15,54	3,51
110	28	60	60	28,04	6,11
235	63	120	120	59,62	12,79
277	71	150	150	69,16	26,15
338	82	183	180	75,55	33,17
361	89	228	220	90,01	105,54
471	119	248	248	115,66	19,05
567	139	300	300	135,18	29,18

Table 9. Comparaison des résultats pour le modèle 4

NF	NP	RS(m)	BAB(m)	RS(t)	BAB(t)
80	24	39	39	24,30	119,29
176	53	80	80	49,30	351,64
284	82	140	140	88,74	341,26
368	107	180	180	121,92	295,80
454	136	220	220	173,37	441,20
525	155	250	250	156,66	1362,06*
617	191	280	280	185,52	441,76
672	209	300	300	198,65	313,96

Les tables 6 à 9 présentent respectivement les résultats numériques obtenus pour les quatre modèles. Nous pouvons constater que le temps de calcul de la méthode de procédure par séparation et évaluation augmente avec le nombre de produits à fabriquer. Cette augmentation n'est pas très rapide en règle générale. Par contre, on peut aussi constater que le temps de calcul augmente très vite avec le nombre total de transitions dans le modèle. Pendant l'expérience, nous avons observé que le temps de calcul augmente exponentiellement avec le nombre de ressources, ce qui n'est pas surprenant, compte tenu du fait que le nombre de nœuds descendants d'un nœud croît exponentiellement avec le nombre de ressources.

Il faut noter que dans certains cas, le temps de calcul peut être très long (ces cas sont signalés par une étoile dans les tables). Ceci est du au fait que la borne inférieure n'est égale à la valeur optimale du critère que lorsque la solution partielle est presque complète. Dans ce cas, il faut parcourir une bonne partie de l'arborescence avant de trouver la solution optimale.

En ce qui concerne la méthode du recuit simulé, le temps de calcul est quasi-proportionel au nombre total de franchsissements des transitions. Comme nous pouvons le constater, la valeur du critère fournie par la méthode du recuit simulé s'éloigne plus de la valeur optimale avec un temps de calcul beaucoup plus important lorsque la taille du problème est petite que lorsque la taille est grande.

En conséquence de ces remarques, nous concluons que lorsque la taille du problème est petite, la procédure par séparation et évaluation est particulièrement efficace. Lorsque la taille du problème est grande, la méthode du recuit simulé devient de plus en plus intéressante.

CONCLUSION

Dans ce rapport, nous avons proposé deux algorithmes ("recuit simulé" et " procédure par séparation et évaluation") pour résoudre le problème d'ordonnancement, en utilisant des modèles de réseaux de Petri. Dans notre modèle, nous prenons en considération l'arrivée des matières premières et le départ des produits finis. Notre modèle tient compte de la polyvalence des ressources, ce qui arrive de plus en plus souvent dans la pratique avec le développement des ateliers flexibles. Les résultats montrent que la méthode par séparation et évaluation est très efficace quand il y a peu de ressources (ce qui est le cas dans les ateliers flexibles). Cependant, avec la NP-difficulté des problèmes d'ordonnancement, elle devient inapplicable lorsque la taille du problème est grande. Dans ce cas, la méthode du recuit simulé semble être une bonne solution.

REFERENCES

[1] CHEN H.X., CHU F., PROTH J.M et SAVI V.M., "A Petri net based production management framework which preserves behavioural qualitative properties", en préparation.

[2] CHRETIENNE P., "Les réseaux de Petri temporisés", Université de Paris VI, Paris, France, Thèse d'Etat, 1983.

[3] COMMONER F., HOLT A., EVEN S. and PNUELI A., "Marked directed graphs", *Journal of Computer and System Science*, Vol. 5, No. 5, 1971.

[4] FRENCH, S. 1982. Sequencing and Scheduling: An introduction to the Mathematics of the Job-Shop. John Wiley & Sons, New York.

[5] HARHALAKIS G., LEVENTOPOULOS M., LIN C.P., NAGI R. and PROTH J.M., "A class conflict free Petri nets used for controlling manufacturing systems", Technical Report No. TR92-90", Systems Research Center, The University of Maryland at College Park, USA, October 1992.

[6] HILLION H.P. and PROTH J.M., "Performance evaluation of job-shop systems using timed event graphs", *IEEE Transactions on Automatic Control*, Vol. 34, No. 1, January 1989.

[7] HOLLOWAY L.E. and KROGH B.H., "Synthesis of feedback control logic for a class of controlled Petri nets", *IEEE Transactions on Automatic Control*, Vol. 35, No. 5, May 1990.

[8] KROGH B.H. and BECK C.L., "Synthesis of place / transition nets for simulation and control of manufacturing systems", Proceedings of IFIP Symp. Large Scale System, Zurich, Switzerland, August 1986.

[9] LAFTIT S., PROTH J.M. and XIE X.L., "Optimization of invariant criteria for event graphs", *IEEE Transactions on Automatic Control*, Vol. 37, No. 5, pp. 547-555, May 1992.

[10] LEE D.Y. and DICESARE F.D., "FMS scheduling using Petri nets and heuristic search", Proceedings of the 1992 IEEE, International Conference on Robotics and Automation, Nice, France - May 1992.

[11] MURATA T., "Petri nets: Properties, analysis and applications", Proceedings IEEE, Vol. 77, No. 4, April 1989.

[12] PETRI C.A., "Communication with Automata", New York: Griffiss Air Force Base. Tech. Rep. RADCTR-65_377, vol. 1,Suppl. 1, 1966.

[13] RAMAMOORTHY C.V. and HO G.S., "Performance evaluation of asynchronous concurrent systems using Petri nets", *IEEE Trans. Software Eng.*, Vol. SE-6, No. 5, pp. 440-449, 1980.

[14] SHIH H M and SEKIGUCHI T., "A timed Petri net and beam search based on-line Fms scheduling system with routing flexibility", Proceedings of the 1991 IEEE, International Conference on Robotics and Automation, Sacramento, California - April 1991.

Unité de Recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Antenne de Metz
Technopôle de Metz 2000 - Cescom - 4, rue Marconi - 57070 METZ (France)

Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399